```
#include "DSP281x_Device.h"



interrupt void TXIsr(void);
interrupt void RXIsr(void);
//interrupt void Cap3ISR(void);

// Initialization Routines

void InitInterrupts()
{
    DINT;                               //Disable Global Interrupts
    IER = 0x0000;                       //Disable CPU interrupts
    IFR = 0x0000;                       //Clear all CPU interrupt flags
    EALLOW;
    PieVectTable.MRINTA = &RXIsr;       //Set name of Recieve ISR
    PieVectTable.MXINTA = &TXIsr;       //Set name of Transmit ISR
//  PieVectTable.CAPINT3 = &Cap3ISR;    //Set name of Cap3 ISR
    EDIS;
    PieCtrlRegs.PIECRTL.bit.ENPIE = 1;  //Enable the PIE block
    PieCtrlRegs.PIEIER6.bit.INTx5=0;    //Disable PIE Group 6, INT 5, McBSP recieve
    PieCtrlRegs.PIEIER6.bit.INTx6=0;    //Enable PIE Group 6, INT 6, McBSP trans
//  PieCtrlRegs.PIEIER3.bit.INTx7=0;    //Enable PIE Group 3, INT 7, Cap 3
    IER=0x0024;                         //Enable CPU INT6 and CPU INT3
    EINT;                               //Enable Global Interrupts
}

void InitMcBSP()
{
    //McBSP Initialization
    //  Note: Some registers set here were not completly neccesary but
    //      were set so that they are at a known state.

    //Reset GPIO Components
    McbspaRegs.SPCR2.bit.FRST=0;    //Reset FS generator
    McbspaRegs.SPCR2.bit.GRST=0;    //Reset Sample rate Generator
    McbspaRegs.SPCR2.bit.XRST=0;    //Reset Transmitter
    McbspaRegs.SPCR1.bit.RRST=0;    //Reset Reciever
                                    //Note: Components are held in the reset state until enabled
    //McBSP Control
    McbspaRegs.SPCR2.all=0x0000;    //Disable Free and Soft modes
    McbspaRegs.SPCR1.all=0x0000;    //Disable digital loop back and A-Bis modes and right justify words
    McbspaRegs.RCR2.all=0x0000;     //Left at default because no second phase
    McbspaRegs.RCR1.all=0x0140;     //Recieve: 1 phase/frame, 2 words/frame, 16 bits/word
    McbspaRegs.XCR2.all=0x0000;     //Left at default because no second phase
    McbspaRegs.XCR1.all=0x0140;     //Transmit: 1 phase/frame, 2 words/frame, 16 bits/word
```

```
    //Multi Channel Control
    McbspaRegs.SRGR2.all=0x0000;    //Leave signal generator at deault settings
    McbspaRegs.SRGR1.all=0x0000;    //Leave signal generator at deault settings
    McbspaRegs.MCR2.all=0x0000;     //Leave at default for single channel opperation
    McbspaRegs.MCR1.all=0x0000;     //Leave at default for single channel opperation

    //Pin Control
    McbspaRegs.PCR.all=0x0080;      //Set all frame and bit clock pins as inputs

    //FIFO Control
    McbspaRegs.MFFTX.all=0x4020;    //Enable TX FIFO Int, Int When FIFO is empty
    McbspaRegs.MFFRX.all=0x0022;    //Enable RX FIFO Int, Int When FIFO is at 2
    McbspaRegs.MFFCT.all=0x0000;    //Only used for SPI
    McbspaRegs.MFFINT.all=0x0000;   //A-Bis Interrupts disabled
    McbspaRegs.MFFST.all=0x0000;    //clear all frame related flags

    //Reenable GPIO Components
    McbspaRegs.MFFTX.bit.TXFIFO_RESET=1;    //Reset and enable TX FIFO
    McbspaRegs.MFFRX.bit.RXFIFO_RESET=1;    //Reset and enable RX FIFO
    McbspaRegs.SPCR1.bit.RRST=1;            //Enable Reciever
    McbspaRegs.SPCR2.bit.XRST=1;            //Enable Transmitter
}

void InitGPIO(void)
{
    //GPIO Initialization
    EALLOW;
    GpioMuxRegs.GPAMUX.all=0x0400;  //Select GPIO for all but McBSP pins
    GpioMuxRegs.GPBMUX.all=0x0000;
    GpioMuxRegs.GPDMUX.all=0x0000;
    GpioMuxRegs.GPFMUX.all=0x3F00;  //Select GPIO pin mode
    GpioMuxRegs.GPEMUX.all=0x0000;
    GpioMuxRegs.GPGMUX.all=0x0000;

    GpioMuxRegs.GPADIR.all=0x83FF;  //Make port a outputs
    GpioMuxRegs.GPBDIR.all=0x0000;
    GpioMuxRegs.GPDDIR.all=0x0000;
    GpioMuxRegs.GPEDIR.all=0x0000;
    GpioMuxRegs.GPFDIR.all=0x4000;  //Make GPIOF14 an output
    GpioMuxRegs.GPGDIR.all=0x0000;
    EDIS;
}

/*

void InitCap(void)
{
```

```
    EvaRegs.CAPCONA.bit.CAPRES = 0;         //Reset capture regs

    EvbRegs.T3CON.bit.TENABLE = 0;          //Enable timer 3
    EvbRegs.T3CON.bit.TMODE = 0x10;         //Continuous count mode
    EvbRegs.T3CON.bit.TCLKS10 = 0;          //Use internal clock
    EvbRegs.T3CON.bit.TENABLE = 1;

    EvaRegs.CAPCONA.bit.CAP3EN = 1;         //Enable Capture 3

    EvaRegs.EVAIMRC.bit.CAP3INT = 1;        //Enable capture 3 interrupt
    EvaRegs.CAPCONA.bit.CAP3TSEL = 1;       //Use GP timer 1(timer 3)
    EvaRegs.CAPCONA.bit.CAP3EDGE = 0x1;     //Trigger on rising edge

}


*/
```